PAST

Scalable Ethernet for Data Centers

Brent Stephens †, Alan Cox †, Wes Felter ‡, Colin Dixon ‡, and John Carter ‡

†Rice University ‡IBM Research

December 11th, 2012

PAST ...

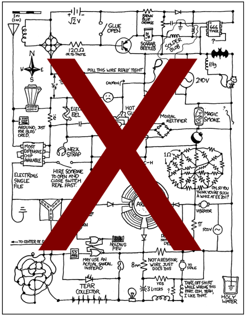
- ...is a large flat L2 network for using arbitrary topologies
- ...is implementable on existing Ethernet switch hardware and unmodified host network stacks
- ...meets or exceeds the performance of the state of the art

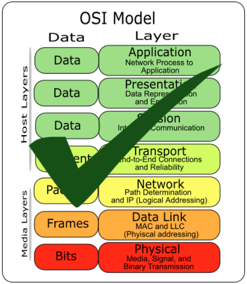
Data Center Network Requirements

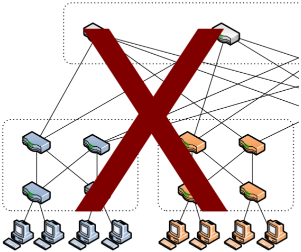
- Host mobility
- Effective use of bandwidth
- Autonomous
- Scalability

Additional Design Requirements

- No hardware changes
- Respects Layering
- Topology Independent



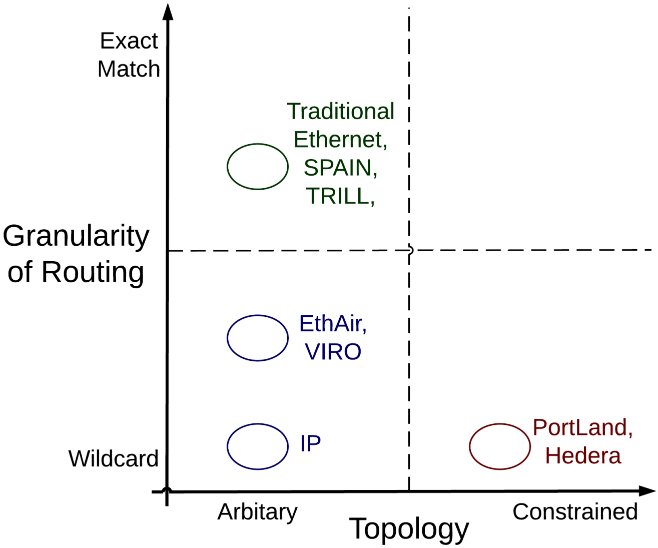




PAST

- *Per-Address Spanning Tree* routing algorithm
- Unmodified Ethernet switches and hosts
  - Implementable today
  - Exploit existing features
- Arbitrary topologies
  - 10's of thousands of hosts
- Performance comparable to or *greater than* ECMP

Routing Space



Routing Space

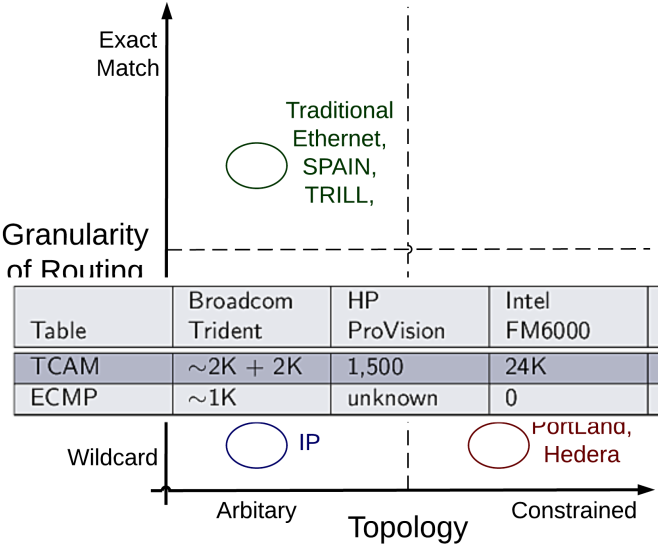


Table	Broadcom Trident	HP ProVision	Intel FM6000	Mellanox SwitchX
TCAM	~2K + 2K	1,500	24K	0?
ECMP	~1K	unknown	0	unknown

Routing Space

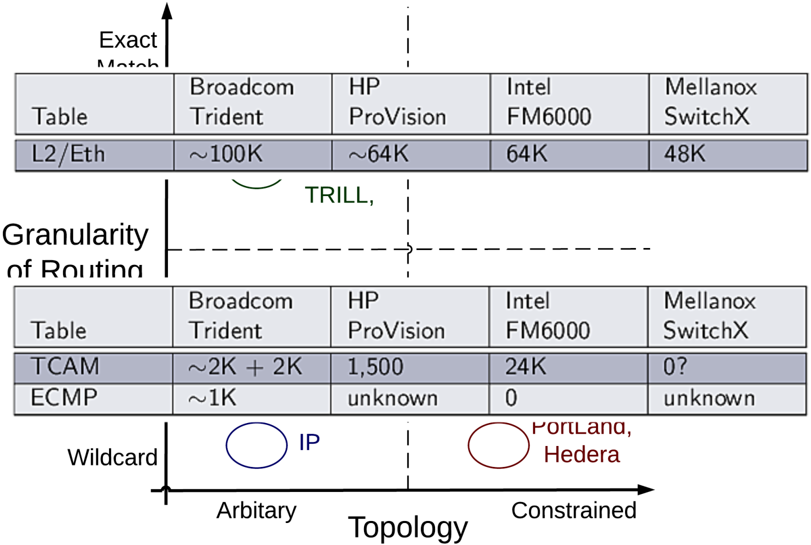
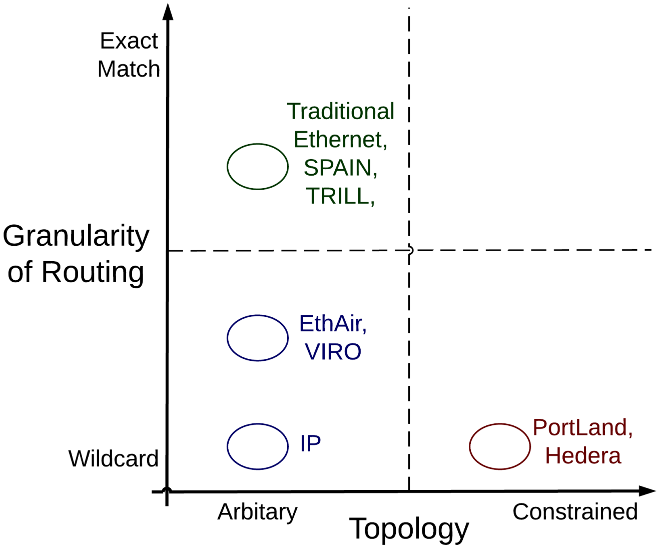


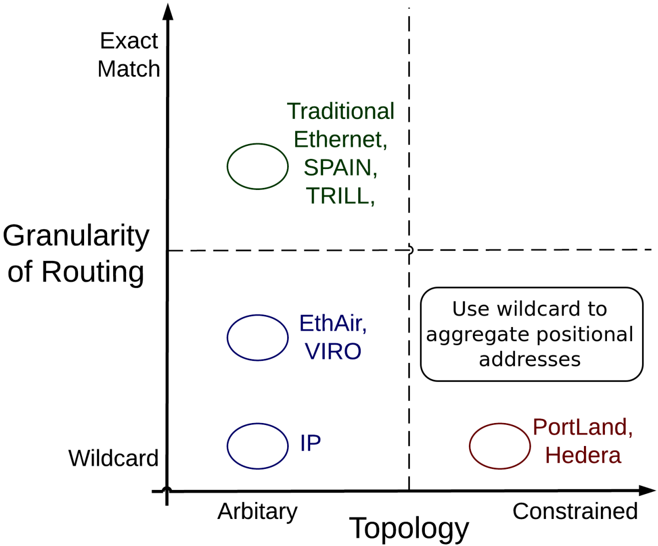
Table	Broadcom Trident	HP ProVision	Intel FM6000	Mellanox SwitchX
L2/Eth	~100K	~64K	64K	48K

Table	Broadcom Trident	HP ProVision	Intel FM6000	Mellanox SwitchX
TCAM	~2K + 2K	1,500	24K	0?
ECMP	~1K	unknown	0	unknown

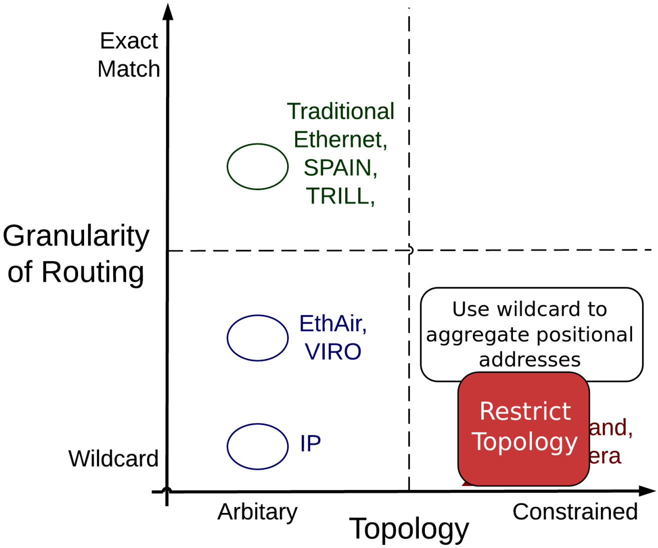
Routing Space



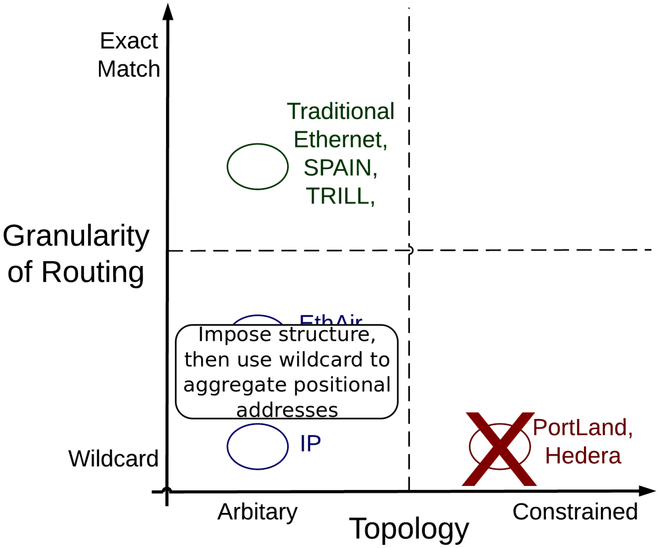
Routing Space



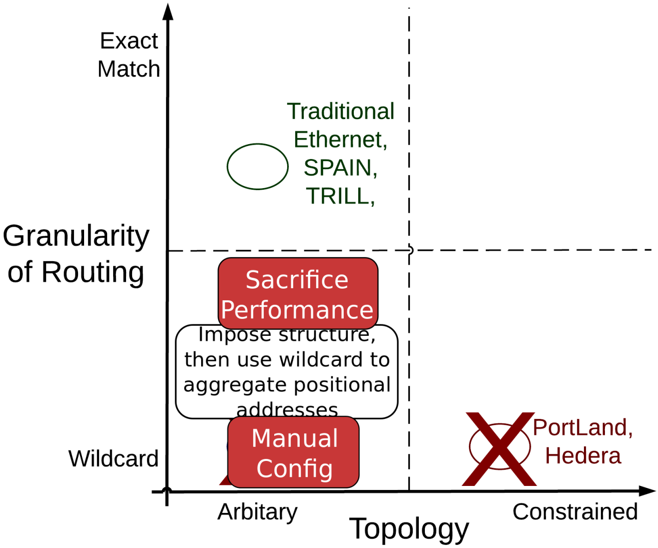
Routing Space



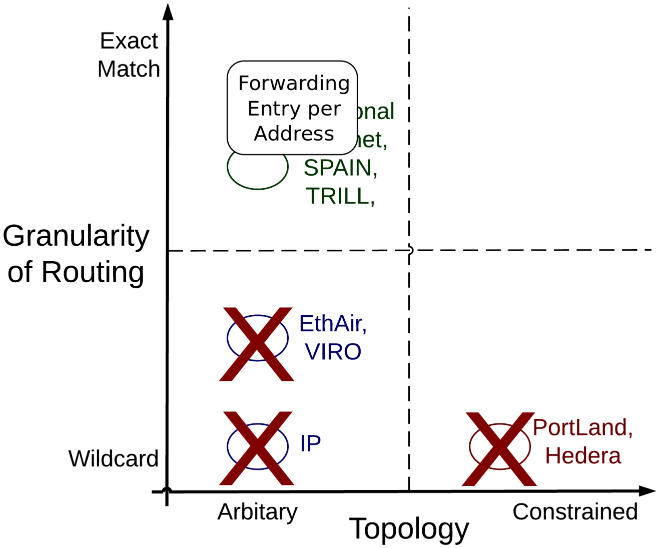
Routing Space



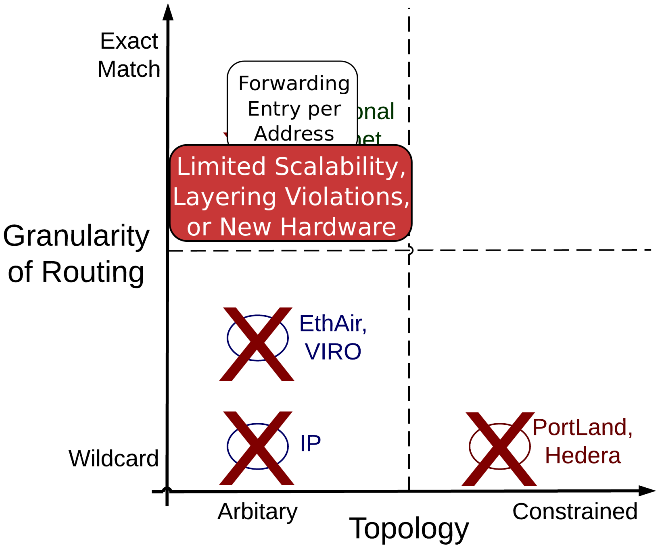
Routing Space



Routing Space



Routing Space



Routing Space

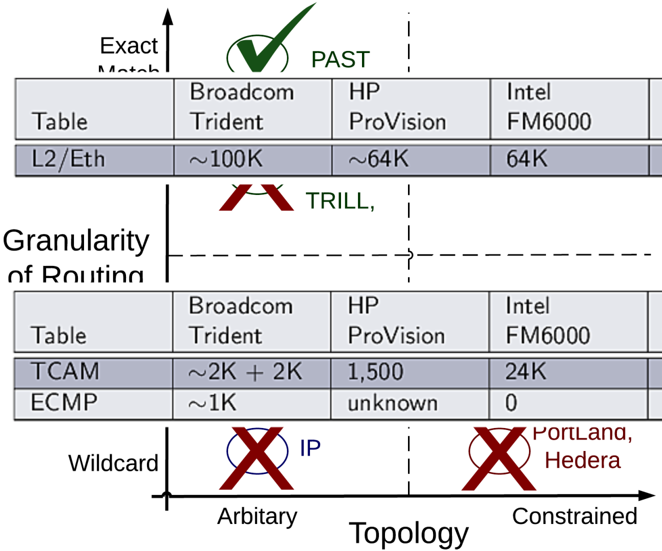
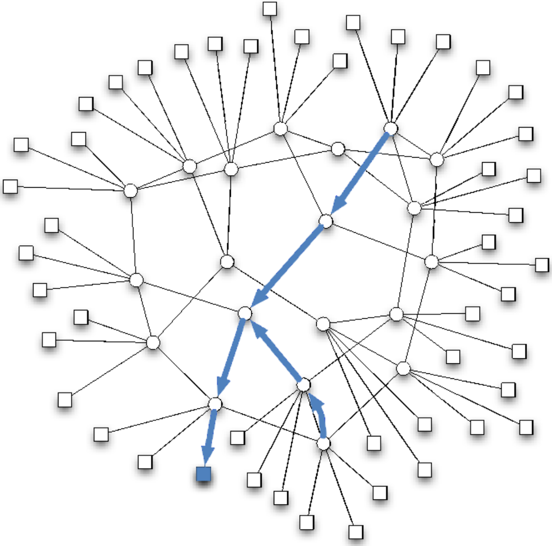


Table	Broadcom Trident	HP ProVision	Intel FM6000	Mellanox SwitchX
L2/Eth	~100K	~64K	64K	48K

Table	Broadcom Trident	HP ProVision	Intel FM6000	Mellanox SwitchX
TCAM	~2K + 2K	1,500	24K	0?
ECMP	~1K	unknown	0	unknown

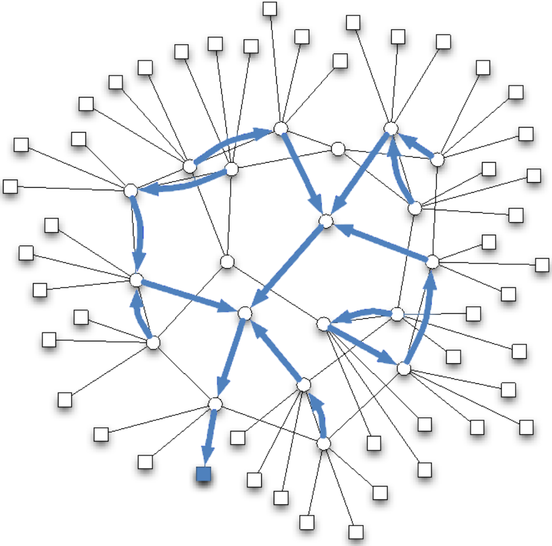
PAST Algorithm

- Goal: Route using the Ethernet table (DMAC, VLAN)
- Constraint 1: Full pair-wise connectivity per-VLAN
- Constraint 2: Ethernet table forces a tree
- Solution: Build a spanning tree rooted at each address
- Load Balances at the *address* ((v-)host) granularity



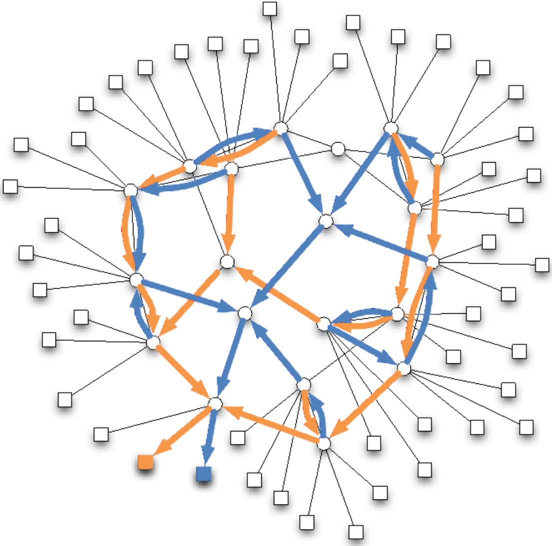
PAST Algorithm

- Goal: Route using the Ethernet table (DMAC, VLAN)
- Constraint 1: Full pair-wise connectivity per-VLAN
- Constraint 2: Ethernet table forces a tree
- Solution: Build a spanning tree rooted at each address
- Load Balances at the *address* ((v-)host) granularity



PAST Algorithm

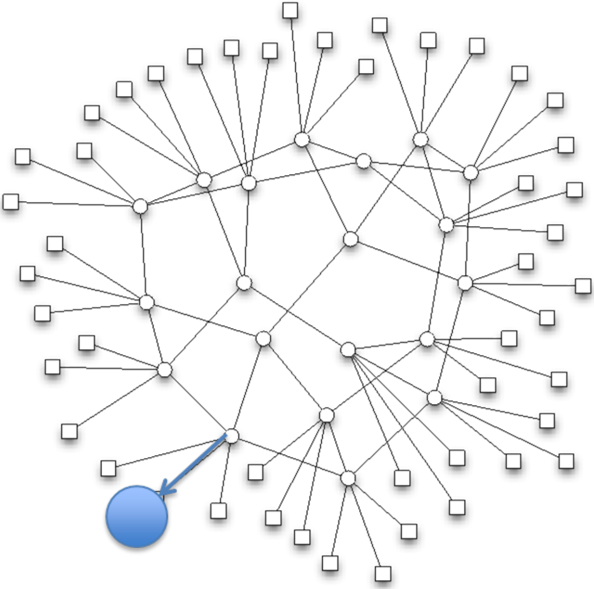
- Goal: Route using the Ethernet table (DMAC, VLAN)
- Constraint 1: Full pair-wise connectivity per-VLAN
- Constraint 2: Ethernet table forces a tree
- Solution: Build a spanning tree rooted at each address
- Load Balances at the *address* ((v-)host) granularity



Tree Construction

- Goal: Use efficient paths
- Solution: Use a BFS tree for minimal paths

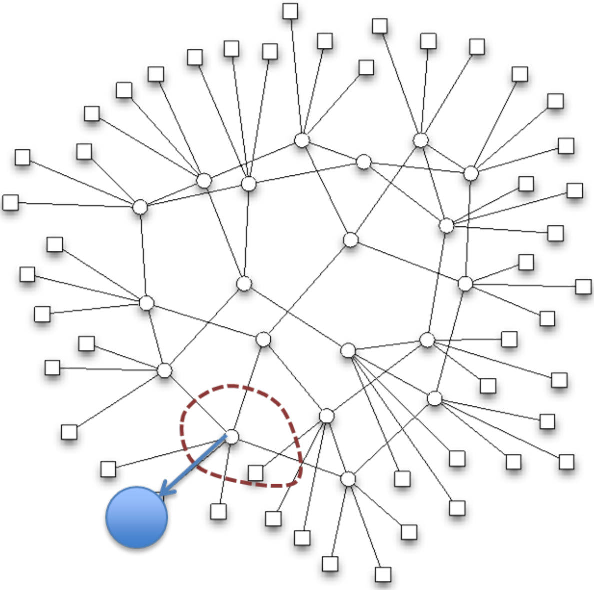
- Goal: Load-balance over all links
- Solution: Tree selection
  - Random
  - Weight links by load



Tree Construction

- Goal: Use efficient paths
- Solution: Use a BFS tree for minimal paths

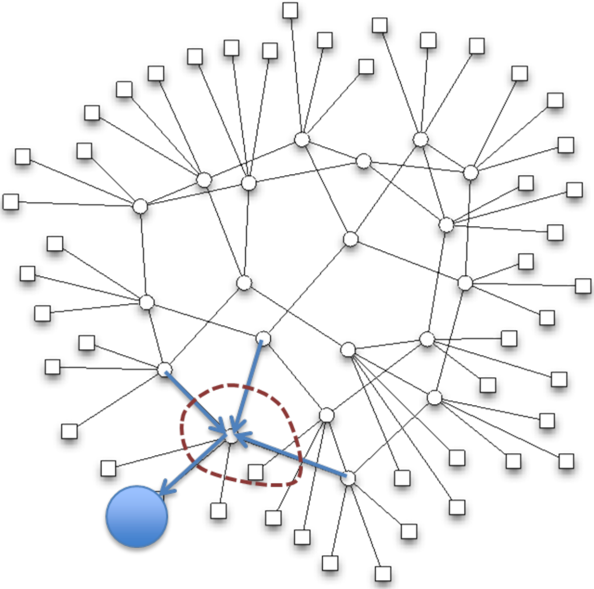
- Goal: Load-balance over all links
- Solution: Tree selection
  - Random
  - Weight links by load



Tree Construction

- Goal: Use efficient paths
- Solution: Use a BFS tree for minimal paths

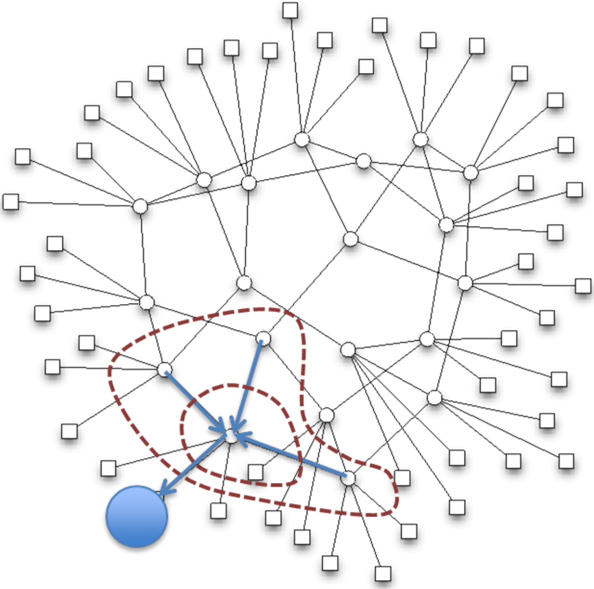
- Goal: Load-balance over all links
- Solution: Tree selection
  - Random
  - Weight links by load



Tree Construction

- Goal: Use efficient paths
- Solution: Use a BFS tree for minimal paths

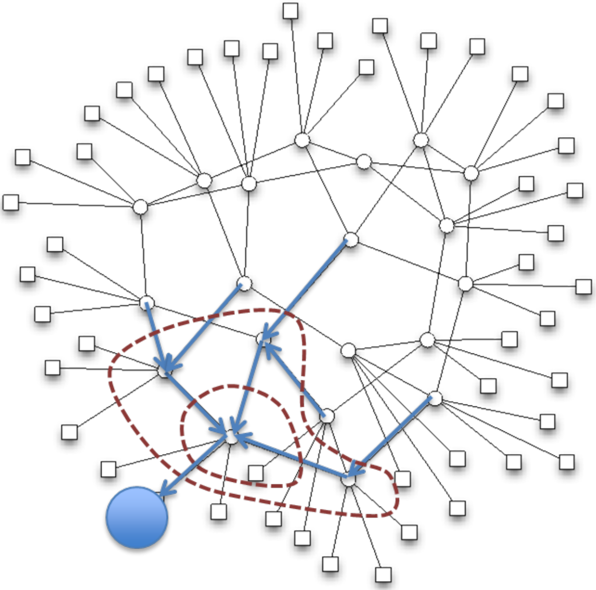
- Goal: Load-balance over all links
- Solution: Tree selection
  - Random
  - Weight links by load



Tree Construction

- Goal: Use efficient paths
- Solution: Use a BFS tree for minimal paths

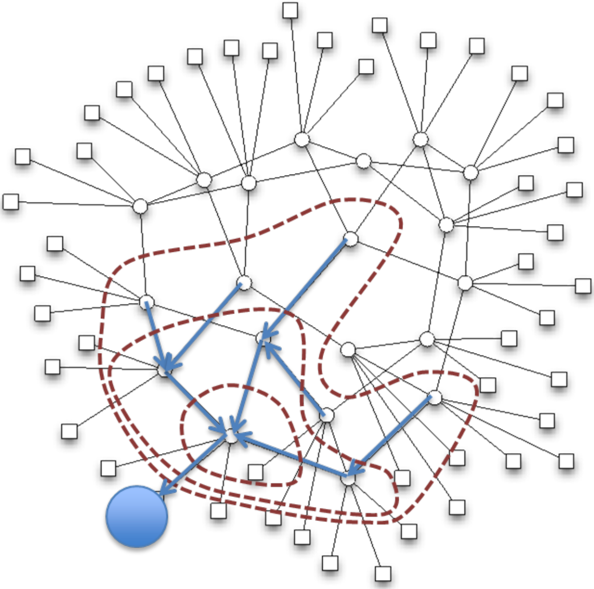
- Goal: Load-balance over all links
- Solution: Tree selection
  - Random
  - Weight links by load



Tree Construction

- Goal: Use efficient paths
- Solution: Use a BFS tree for minimal paths

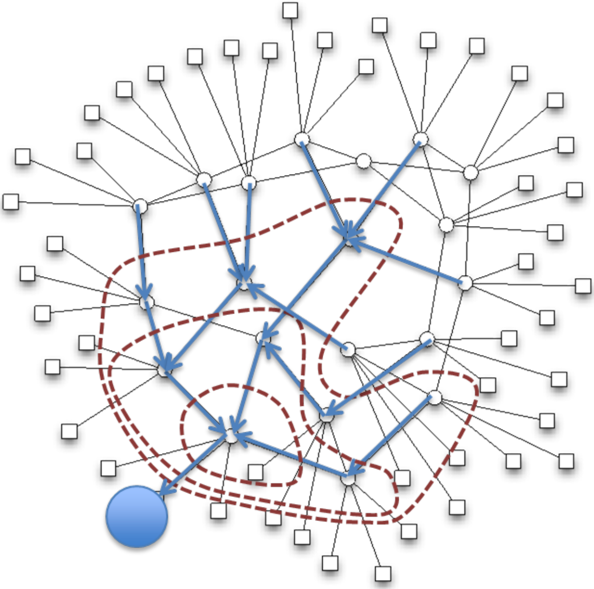
- Goal: Load-balance over all links
- Solution: Tree selection
  - Random
  - Weight links by load



Tree Construction

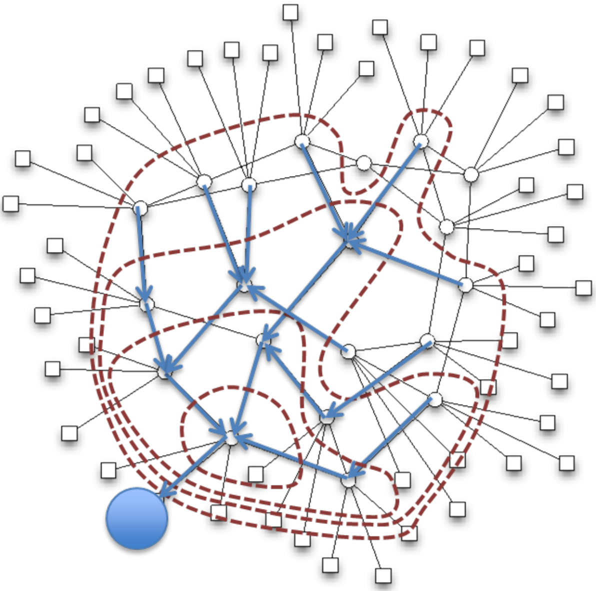
- Goal: Use efficient paths
- Solution: Use a BFS tree for minimal paths

- Goal: Load-balance over all links
- Solution: Tree selection
  - Random
  - Weight links by load



Tree Construction

- Goal: Use efficient paths
- Solution: Use a BFS tree for minimal paths

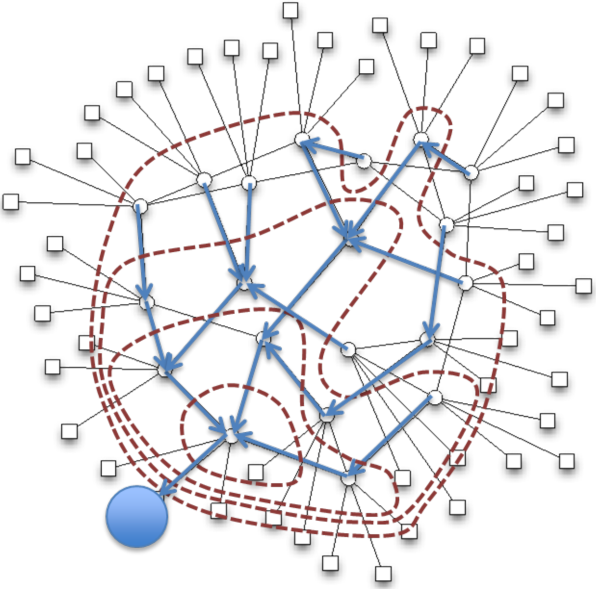


- Goal: Load-balance over all links
- Solution: Tree selection
  - Random
  - Weight links by load

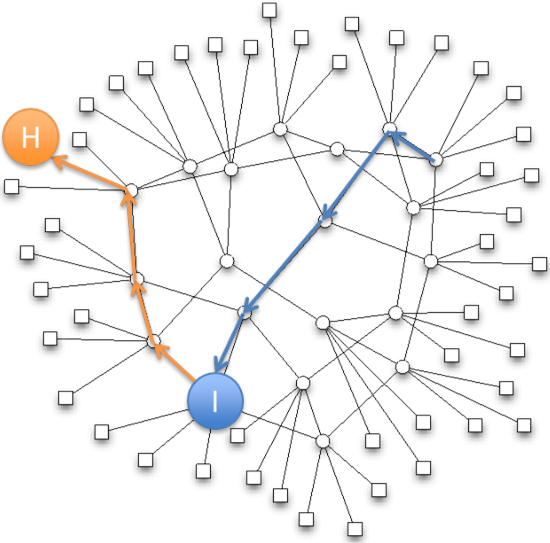
Tree Construction

- Goal: Use efficient paths
- Solution: Use a BFS tree for minimal paths

- Goal: Load-balance over all links
- Solution: Tree selection
  - Random
  - Weight links by load

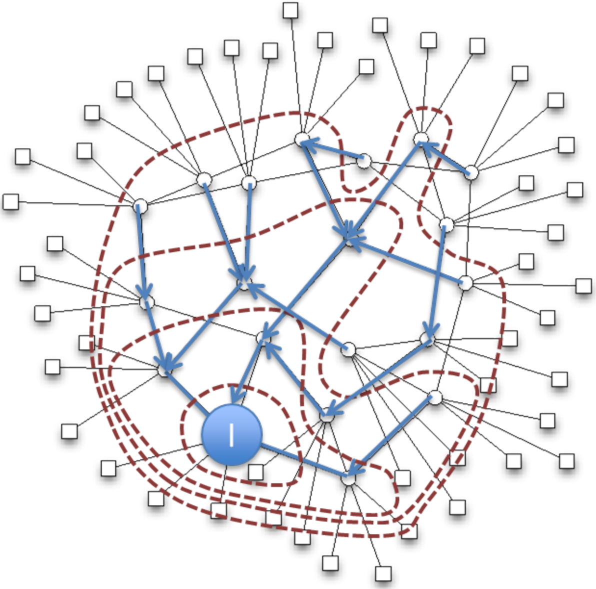


Valiant Load Balancing



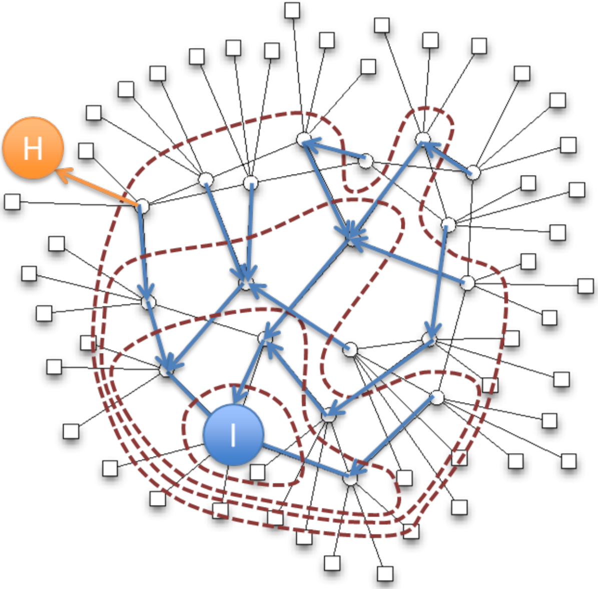
Non-Minimal Tree Construction

- NM-PAST
  - Root the tree for host *h* at a random intermediate switch *i*
  - Inspired by Valiant Load Balancing



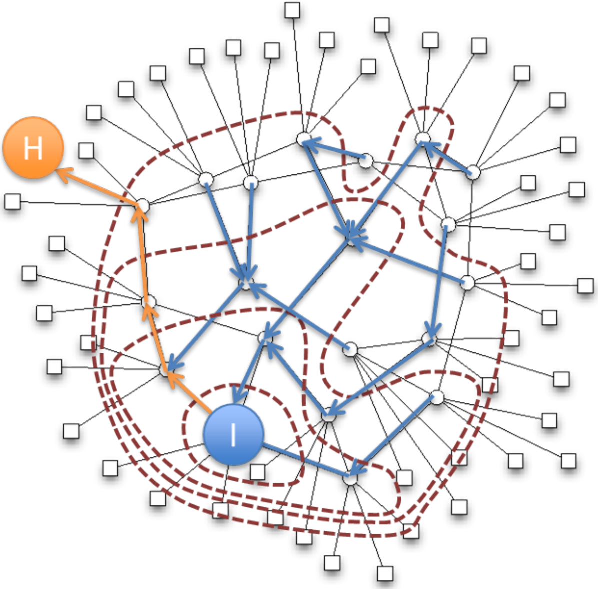
Non-Minimal Tree Construction

- NM-PAST
  - Root the tree for host *h* at a random intermediate switch *i*
  - Inspired by Valiant Load Balancing



Non-Minimal Tree Construction

- NM-PAST
  - Root the tree for host *h* at a random intermediate switch *i*
  - Inspired by Valiant Load Balancing



PAST Discussion

Broadcast/Multicast

Unaffected. May be provided through STP or SDN

Security

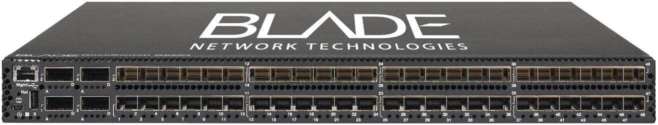
Use VLANs as normal

Virtualization

Use any higher layer virtualization overlay (NetLord, SecondNet, MOOSE, VXLAN)

PAST Implementation

IBM RackSwitch G8264



Implementation Scalability

Eliminate Broadcasts

Improve scalability by using controller for address detection and resolution (ARP, DHCP, IPv6 ND, and RS)

Route Computation

8,000 hosts $\Rightarrow 40\mu s - 1ms$ per tree (300ms per network)

Trivially Parallelizable

Route Installation

Install and forward to 100K addresses

2-12ms rule install latency $\Rightarrow$ masked by migration latency

Failure Recovery

Should patch affected portions of trees

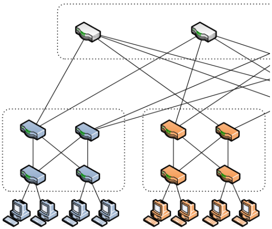
Simulator

- Simulate to evaluate performance at scale
  - Flow based simulator assumes max-min fairness
- Workloads

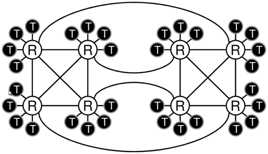
URand-8	Stride-64	Shuffle-10	MSR
$i \in 1..8$ $Dst^i = rand()\%N$, *Benign*	$Dst_n = (n + 64)\%N$, *Adversarial*	128MB to all hosts, Random order, 10 active connections, *More stressful than URand*	Synthetically generated from 1500-server cluster, *Light load*

Topologies

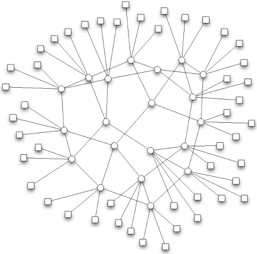
- Compare equal bisection-bandwidth (oversubscription ratio) networks



EGFT
(Fat Tree)



HyperX
(Flattened Butterfly)



Jellyfish
(Random Regular Graph)

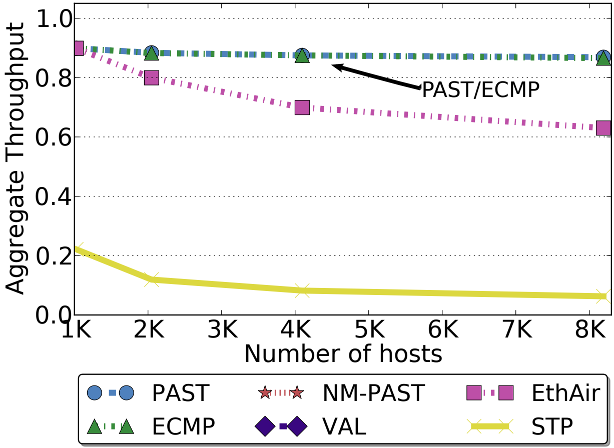
Evaluation

- Demonstrate PAST performance equal to or greater than other routing algorithms
- Demonstrate PAST performs well under adversarial workloads
- Demonstrate that PAST can effectively use a variety of topologies

Evaluation

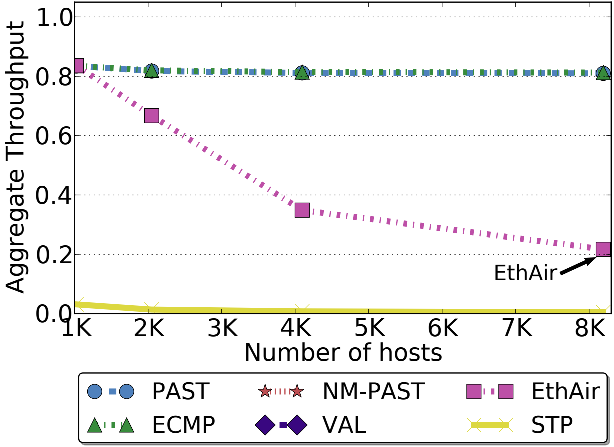
- Demonstrate PAST performance equal to or greater than other routing algorithms
  - URand-8 on a 1:2 Bisection Bandwidth HyperX
  - Shuffle-10 on a 1:2 Bisection Bandwidth HyperX
- Demonstrate PAST performs well under adversarial workloads
- Demonstrate that PAST can effectively use a variety of topologies

PAST matches ECMP



Shuffle-10 on a 1:2 Bisection Bandwidth HyperX

EthAir scales poorly



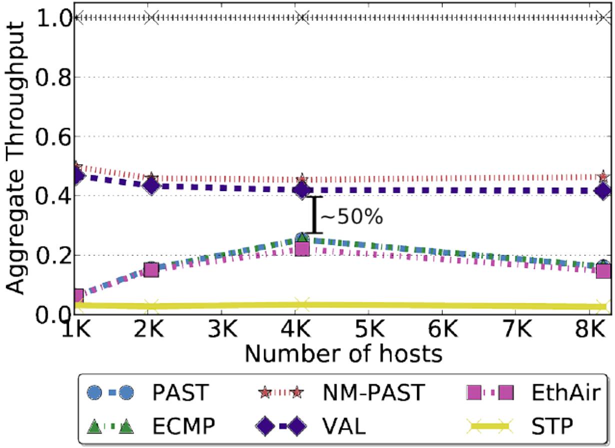
Evaluation

- Demonstrate PAST performance equal to or greater than other routing algorithms
  - PAST matches ECMP
  - EthAir scales poorly
- Demonstrate PAST performs well under adversarial workloads
- Demonstrate that PAST can effectively use a variety of topologies

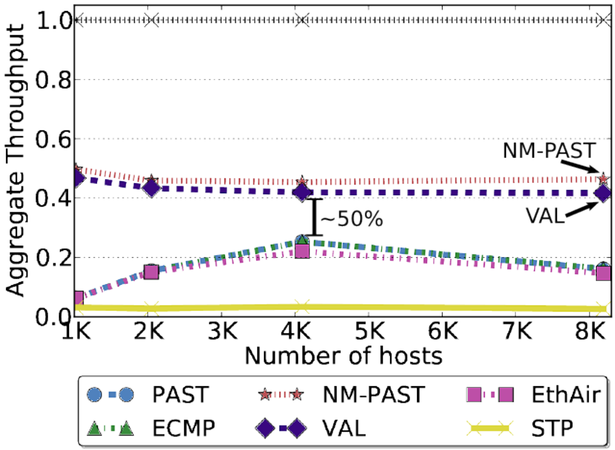
Evaluation

- Demonstrate PAST performance equal to or greater than other routing algorithms
- Demonstrate PAST performs well under adversarial workloads
  - Stride-64 on a 1:1 Bisection Bandwidth HyperX
  - Shuffle-10 on a 1:2 Bisection Bandwidth HyperX
- Demonstrate that PAST can effectively use a variety of topologies

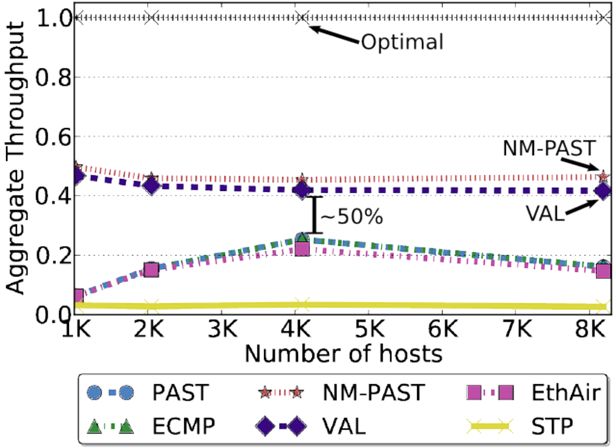
NM-PAST can double performance . . .



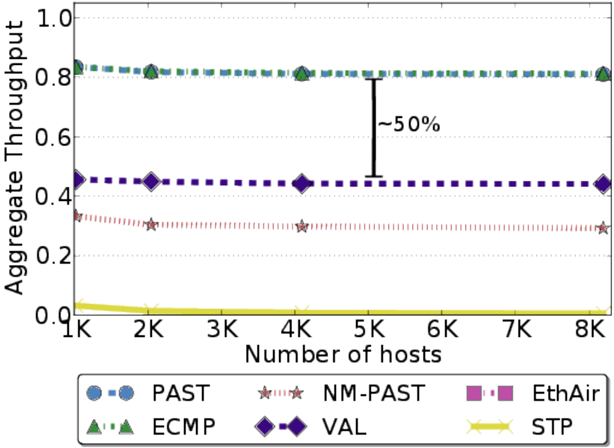
. . . and NM-PAST matches VAL . . .



...although collisions can hurt performance



VAL halves performance under uniform workloads



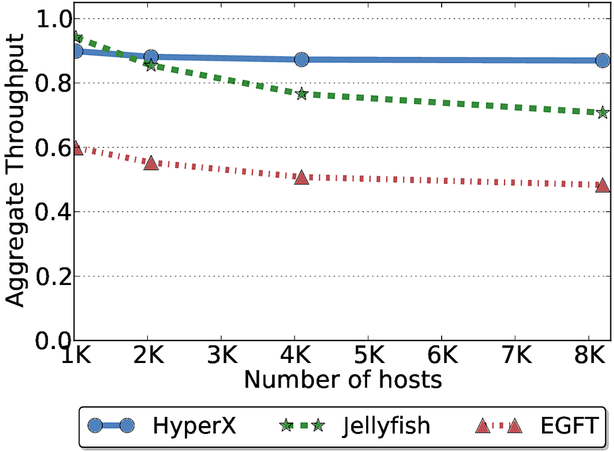
Evaluation

- Demonstrate PAST performance equal to or greater than other routing algorithms
- Demonstrate PAST performs well under adversarial workloads
  - NM-PAST can double performance
  - NM-PAST matches VAL
  - NM-PAST and VAL halve performance under uniform workloads
- Demonstrate that PAST can effectively use a variety of topologies

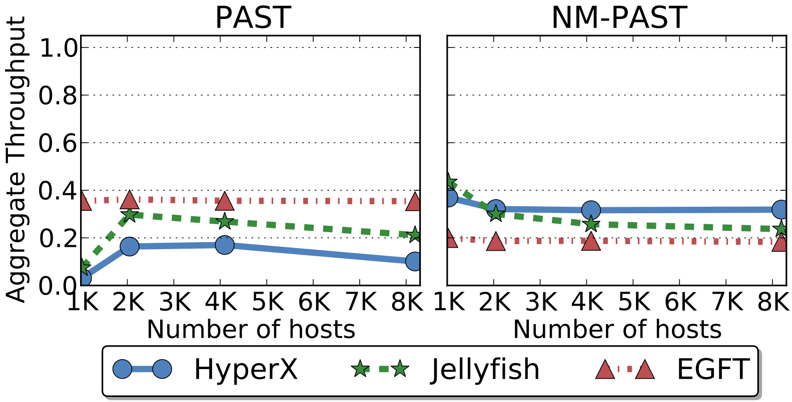
Evaluation

- Demonstrate PAST performance equal to or greater than other routing algorithms
- Demonstrate PAST performs well under adversarial workloads
- Demonstrate that PAST can effectively use a variety of topologies
  - URand-8 on 1:2 Bisection Bandwidth networks
  - Stride-64 on 1:2 Bisection Bandwidth networks

PAST on HyperX and Jellyfish outperforms EGFT



NM-PAST on a HyperX matches PAST on an EGFT



Evaluation

- Demonstrate PAST performance equal to or greater than other routing algorithms
- Demonstrate PAST performs well under adversarial workloads
- Demonstrate that PAST can effectively use a variety of topologies
  - PAST on HyperX and Jellyfish outperforms EGFT
  - NM-PAST enables HyperX to perform well under adversarial workloads

Conclusions

- PAST is a datacenter network that supports full host mobility, high bandwidth, self-configuration, and tens of thousands of hosts
- PAST can provide near optimal throughput
  - Worst case performance equal to ECMP
  - Best case performance double ECMP
  - ECMP is not as useful (or necessary) as previously thought
- PAST supports commodity switches and exploits only the most basic Ethernet hardware
  - PAST is implementable today